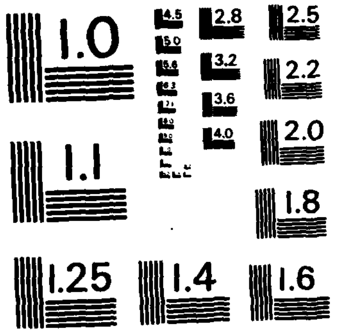
AD-A139 552 SOLUTION TECHNIQUES IN FINITE ELEMENT ANALYSIS(U) 1/1
CALIFORNIA UNIV BERKELEY DEPT OF CIVIL ENGINEERING
B NOUR-OMID ET AL. MAY 83 NCEL-CR-83.027
UNCLASSIFIED N62474-82-C-8277 F/G 20/11 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(1)

CR 83.027

NAVAL CIVIL ENGINEERING LABORATORY
Port Hueneme, California

Sponsored by
NAVAL FACILITIES ENGINEERING COMMAND

SOLUTION TECHNIQUES IN FINITE ELEMENT ANALYSIS

May 1983

An Investigation Conducted by
UNIVERSITY OF CALIFORNIA
Department of Civil Engineering
Berkeley, California

N62474-82-C-8277

METRIC CONVERSION FACTORS

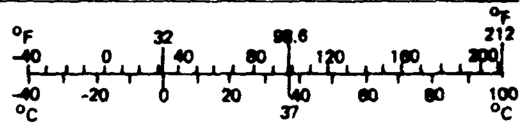
Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2,000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.

Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1,000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER CR 83.027	2 GOVT ACCESSION NO. AD-A139	3 RECIPIENT'S CATALOG NUMBER 552
4 TITLE (and Subtitle) SOLUTION TECHNIQUES IN FINITE ELEMENT ANALYSIS		5 TYPE OF REPORT & PERIOD COVERED Not Final Mar 1982 - Jan 1983 6 PERFORMING ORG REPORT NUMBER
7 AUTHOR(s) Bahram Nour-Omid, Carlos Rodrigues, and Robert L. Taylor		8 CONTRACT OR GRANT NUMBER(s) N62474-82-C-8277
9 PERFORMING ORGANIZATION NAME AND ADDRESS UNIVERSITY OF CALIFORNIA Department of Civil Engineering Berkeley, Calif.		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS YR023.03.01.003
11 CONTROLLING OFFICE NAME AND ADDRESS Naval Civil Engineering Laboratory Port Hueneme, Calif.		12 REPORT DATE May 1983 13 NUMBER OF PAGES ~~29~~ 32
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Facilities Engineering Command 200 Stovall Street Alexandria, Va.		15 SECURITY CLASS (of this report) Unclassified 15a DECLASSIFICATION DOWNGRADING SCHEDULE
16 DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18 SUPPLEMENTARY NOTES		
19 KEY WORDS (Continue on reverse side if necessary and identify by block number) Structural mechanics; finite elements; nonlinear algebraic equations; numerical solution methods		

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

A desirable advantage of iterative methods is that they provide means of controlling the accuracy of the solution. In particular, when low levels of accuracy are required this can result in faster algorithms than the direct methods. The use of the conjugate gradient algorithm to solve the linearized system of equations is considered. A preconditioning matrix based on a

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

splitting method is constructed. The outcome is an algorithm which results in substantial reduction in storage over direct methods. The above method is compared with its rivals on several quite different problems in structural mechanics and favorable results were obtained.

TABLE OF CONTENTS

1. Introduction

The finite element method of discretization is used to reduce many complex continuum problems to discrete systems. Although this reduction is the most important step in the overall analysis of a structure, solving the discrete problem is far from trivial. In general, the reduced system is nonlinear and an iterative method must be employed to arrive at the solution. Most solution methods are based on some form of Newton's method in which the nonlinear problem is linearized using an initial approximation to arrive at a linear set of simultaneous equations. The solution of the set of linear equations leads to a correction of the initial approximation. When solving the linear equations, one should not loose sight of the primary objective: solving the nonlinear problem.

Iterative methods, such as the conjugate gradient or Lanczos method, are among the many methods that may be used to solve systems of linear equations. The advantage of these methods, when used as the inner loop of the Newton iteration, is twofold.

(i) The linear equation may be solved to any desired level of accuracy as governed by the Newton iteration.

(ii) A considerable reduction in storage can be achieved when no triangular factorization need be performed.

In [4] a method was developed, based on the preconditioned Lanczos method, to realize some of the advantages of iterative methods. In this previous study, the triangular factors of the initial tangent matrix were used to form a preconditioning matrix for the subsequent solution steps. In the present we have eliminated factorizations by employing other preconditioners and further, have reduced the storage needs of the method.

2. A Preconditioned Conjugate Gradient Method

An essential step in nonlinear analysis of structures is solving a linear system of algebraic equations. The preconditioned conjugate gradient method (hereafter called PCG) is one of the many procedures for solving

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} = 0 \tag{2.1}$$

where $\mathbf{A}$ is an $n \times n$ symmetric positive definite matrix (which is sparsely populated) and $\mathbf{b}$ is the right-hand side vector. In the case of static analysis, $\mathbf{A}$ is the current tangent matrix and in the case of dynamic analysis, $\mathbf{A}$ depends on the mass, damping and stiffness matrices, as well as the time increment.

The initial popularity of the conjugate gradient method was due to a number of factors. In exact arithmetic the method required a maximum of n iterations to solve (2.1) which made the method superior to other iterative methods. In fact conjugate gradient is in the class of semi-iterative methods which also includes the Lanczos algorithm [6]. The disadvantage of direct methods is their large storage demands for keeping the factors of $\mathbf{A}$. The only interface between the conjugate gradient method and $\mathbf{A}$ is through the product $\mathbf{A}\mathbf{v}$ for a given vector $\mathbf{v}$. This is an elegant way of taking advantage of sparsity of $\mathbf{A}$ which has the added advantage that $\mathbf{A}$ need not be known explicitly but only a means of computing the matrix-vector product is required.

The popularity of the conjugate gradient method vanished once it was found that under certain conditions the method required as many as $5n$ or $6n$ steps to reduce the residual to the desired level. This blemish is accounted for by the strong influence of round-off error.

The addition of preconditioning eliminated this difficulty. Instead of solving (2.1) we solve

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \tag{2.2}$$

for some appropriate choice of $\mathbf{P}$. The object then is to choose $\mathbf{P}$ such that the coefficient matrix of (2.2) is well conditioned.

Theoretical considerations suggest that at the end of each iteration of CG the residual norm is reduced by a factor $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ when solving (2.1) where κ is the condition number of $\mathbf{A}$, defined by $\kappa = \|\mathbf{A}\| \; \|\mathbf{A}^{-1}\|$. See [1] for more details. Note that when $\kappa = 1$, one iteration is sufficient to solve the equation. This provides us with a guideline for choosing $\mathbf{P}$. For a well chosen $\mathbf{P}$ only a few iterations reduce the residual norm to the desired level. Here we give an outline for the preconditioned conjugate algorithm:

Given an initial guess $\mathbf{x}_0$, a positive definite preconditioning matrix $\mathbf{P}$, the matrix $\mathbf{A}$ and the right hand side $\mathbf{b}$:

```
(1) Set $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
(2) Solve $\mathbf{P}\mathbf{d}_0 = \mathbf{r}_0$, for $\mathbf{d}_0$
(3) for $k = 0, 1, 2, \cdots$ until convergence do
    (a) $\alpha_k = (\mathbf{r}_k, \mathbf{d}_k) / (\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)$
    (b) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
    (c) $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
    (d) Solve $\mathbf{P}\mathbf{d}_{k+1} = \mathbf{r}_{k+1}$
    (e) $\beta_k = (\mathbf{r}_{k+1}, \mathbf{d}_{k+1}) / (\mathbf{r}_k, \mathbf{d}_k)$
    (f) $\mathbf{p}_{k+1} = \mathbf{d}_{k+1} + \beta_k \mathbf{p}_k$
```

The operation $(\mathbf{v}, \mathbf{u})$ denotes the inner product $\mathbf{v}^T\mathbf{u}$. The algorithm generates a sequence of approximations to the solution $\mathbf{x}$ with a corresponding residual vector $\mathbf{r}_k$. The termination criterion can be chosen based on these quantities. In addition to storage demands for $\mathbf{A}$ and $\mathbf{P}$ the algorithm requires storage for 4 vectors. The total number of operation per iteration is $NZA + NZP + 5N$, where NZA and NZM are the number of operations for forming $\mathbf{Au}$ and $\mathbf{P}^{-1}\mathbf{v}$ for a given $\mathbf{u}$ and $\mathbf{v}$.

3. Splitting Methods

Here we turn to a topic which at first sight may seem unrelated to the solution of nonlinear algebraic equations. Consider the system of first order differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{3.1}$$

where $\mathbf{x}$ is an n-dimensional vector, the superposed dot, ($\dot{}$), denotes differentiation with respect to time and $\mathbf{f}$ is a function of the unknown vector $\mathbf{x}$ and t.

We consider a special form of $\mathbf{f}$ which can be written as a sum of its subcomponents $\mathbf{f}_i$.

$$\mathbf{f} = \sum_{i=1}^{s} \mathbf{f}_i \tag{3.2}$$

Under these conditions the original problem can be thought as a sum of s sub-problems

$$\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}, t) \quad i = 1, \ldots, s \tag{3.3}$$

In the case of finite element discretization of the spatial domain the sum in (3.2) ranges over the elements. In other cases the splitting may be formed by other means, one of which is demonstrated in the following section.

A consistent algorithm for the solution of (3.1), based on the notion of a splitting technique [3], can now be constructed as a product of algorithms for the sub-problems. In other words, write the algorithm for (3.3) as

$$\mathbf{x}_{m+1} = S_i^{(h)}[\mathbf{x}_m] \tag{3.4}$$

where $S_i^{(h)}$ is an operator denoting the algorithm and the index m ranges over the increment in time, h. Then the algorithm for (3.1) can be written as

$$\mathbf{x}_{m+1} = S^{(h)}[\mathbf{x}_m] \tag{3.5}$$

where

$$S^{(h)} = \prod_{i=1}^{s} S_i^{(h)} \tag{3.6}$$

One of the disadvantages of the splitting method is its low accuracy. The best that these methods can achieve is second order accuracy. That is the truncation error is of the order of h^3 at best. In the sequel we will use the above procedure to construct a preconditioning matrix for the conjugate gradient algorithm described in section 2. The inherent inaccuracy of the splitting method poses no problem since the algorithm is used only as a preconditioner and therefore one can obtain very high accuracies through the conjugate gradient iteration.

4. Solution of Static Problems

Consider the system of linear first order differential equations

$$\tau \dot{\mathbf{x}} + \mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.1}$$

where τ is a given parameter. Formally the solution to equation (4.1) is

$$\mathbf{x}(t) = e^{-\mathbf{A}t/\tau}(\mathbf{x}_0 - \mathbf{A}^{-1}\mathbf{b}) + \mathbf{A}^{-1}\mathbf{b} \tag{4.2}$$

where $\mathbf{x}_0 = \mathbf{x}(0)$, is an initial condition. We observe from (4.2) thet as t ends to infinity $\mathbf{x}(t)$ converges to the solution of (2.1) for $\tau > 0$. Consiquently (4.1) may be utilised to solve the linear equations (2.1). Indeed this approach has been suggested previously (e.g., see [5]). In general the exponential of a large matrix cannot be easily computed and a numerical solution of (4.1) must be used. In order to achieve a soluion of (2.1) a numerical solution to (4.1) must be assymptotically correct for infinite Δt, or a very large number of time steps must be used to compute the solution at infinite time. Here we are not concerned with constructing accurate solution to (4.1) rather we consider the method as a means of constructing a suitable preconditioning matrix for the conjugate gradient algorithm described above.

Splitting methods may be applied to any problem of the form

$$\dot{\mathbf{x}} = \mathbf{B}\mathbf{x} \tag{4.3}$$

where $\mathbf{B}$ is an additive operator defined by

$$\mathbf{B} = \sum_{i=1}^{s} \mathbf{B}_i \tag{4.4}$$

such that the equations

$$\dot{\mathbf{x}} = \mathbf{B}_i \mathbf{x} \qquad i = 1,\ldots,s \tag{4.5}$$

are significantly easier to solve than the original equations. The time stepping algorithm for the global problem is then the product of all the time stepping algorithms for the subproblems with a fractional time step h/s [3].

The coefficient matrix $\mathbf{A}$ in (2.1) may be written as the sum of its diagonal matrix, $\mathbf{D}$, a strictly lower triangular matrix, $\mathbf{L}$, and a srictly upper triangular matrix such that

$$\mathbf{A} = (\tfrac{1}{2}\mathbf{D} + \mathbf{L}) + (\tfrac{1}{2}\mathbf{D} + \mathbf{L})^T \tag{4.6}$$

The associated subproblems, $\dot{\mathbf{x}} = -(\tfrac{1}{2}\mathbf{D} + \mathbf{L})\mathbf{x}$ and $\dot{\mathbf{x}} = -(\tfrac{1}{2}\mathbf{D} + \mathbf{L}^T)\mathbf{x}$ can be solved easily. Applying a backward difference method with a time step $h/2$ to each of the subproblems, we arrive at

$$\mathbf{x}_{m+1} = \left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}\right]^{-1}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}^T)\right]^{-1}\left[\frac{h}{2\tau}\mathbf{b} + \mathbf{x}_m\right] \tag{4.7}$$

where $\mathbf{x}_m$ is an approximation to $\mathbf{x}(mh)$. For initial condition $\mathbf{x}_0 = 0$ we get an approximation to $\mathbf{x}(h)$

$$\mathbf{x}_1 = \left\{\frac{h}{2\tau}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L})\right]^{-1}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}^T)\right]^{-1}\right\}\mathbf{b} \tag{4.8}$$

which is compared to the exact solution

$$\mathbf{x}(h) = \left(\mathbf{I} - e^{-\mathbf{A}h/\tau}\right)\mathbf{A}^{-1}\,\mathbf{b} \tag{4.9}$$

Comparing equations (4.8) and (4.9) suggests that the coefficient matrix in (4.8) may be a *good approximation to* $\mathbf{A}^{-1}$ *for large* h, and may therefore be an effective preconditioning matrix. The scalar factor $\frac{h}{2\tau}$ may be ignored for preconditioning purposes.

When using this in conjunction with conjugate gradient algorithm of section 2 the preconditioning matrix becomes

$$\mathbf{P} = (\mathbf{I} + \omega/2\mathbf{D} + \omega\mathbf{L})(\mathbf{I} + \omega/2\mathbf{D} + \omega\mathbf{L}^T) \tag{4.10}$$

where $\omega = h/2\tau$ is now a free parameter.

To simplify the choice of ω we scale the stiffness matrix $\mathbf{A}$ such that diagonal of $\mathbf{A}$ is unity. The resulting matrix is $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. The system of equations (2.1) now becomes

$$\bar{\mathbf{A}}\,\bar{\mathbf{x}} = \bar{\mathbf{b}} \tag{4.11}$$

where $\bar{\mathbf{x}} = \mathbf{D}^{1/2}\mathbf{x}$ and $\bar{\mathbf{b}} = \mathbf{D}^{-1/2}\mathbf{b}$.

The preconditioned matrix must now be applied to (4.11) resulting in

$$\tilde{\mathbf{P}} = (\mathbf{I} + \omega \tilde{\mathbf{L}})(\mathbf{I} + \omega \tilde{\mathbf{L}}^T) \qquad (4.12)$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{L}} + \tilde{\mathbf{L}}^T$. It is easy to show that preconditioning (4.11) using $\tilde{\mathbf{P}}$ is equivalent to preconditioning (2.1) with

$$\mathbf{P} = (\mathbf{D} + \omega \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \omega \mathbf{L}^T) \qquad (4.13)$$

This can be identified as a member of the class of incomplete Choleski preconditioners [2]. Note that when $\omega = 0$, $\mathbf{P}$ becomes the diagonal matrix $\mathbf{D}$, resulting in the simplest form of preconditioning; diagonal scaling. When $\omega = 1$ then $\mathbf{P} = \mathbf{A} + \mathbf{L}\mathbf{D}^{-1}\mathbf{L}^T$. The error matrix $\mathbf{L}\mathbf{D}^{-1}\mathbf{L}^T$ is rank deficient since $\mathbf{L}$ has zero diagonals. If the norm of $\mathbf{D}$ is larger then the norm of $\mathbf{L}$ then the norm of the error matrix will be small compared to the norm of $\mathbf{A}$ consequently, for most problems it is expected that the optimum ω will be close to unity.

5. Solution of Dynamic Problems

We construct a preconditioning matrix for the linear system of equations arising in a step-by-step algorithm for dynamic analysis of linear and nonlinear structures. In particular, we consider the Newmark algorithm and the preconditioning matrix follows from the splitting method of section 3, in much the same way as for the static problem.

Consider the linear equations of motion

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{5.1}$$

where $\mathbf{M}$ is the diagonal mass matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{f}$ is the external load vector and $\mathbf{u}$ is the response of the structure. For simplicity, we ignore the damping effects, but all the following results may be extended easily to the damped case. The linear system of equations arising at every time step of the Newmark method is

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.2}$$

where

$$\mathbf{A} = \mathbf{K} + \frac{1}{\beta \Delta t^2}\mathbf{M} \tag{5.3}$$

and

$$\mathbf{b} = \mathbf{f}_{t+\Delta t} + \frac{1}{\beta \Delta t^2}\mathbf{M}[\mathbf{u}_t + \Delta t\,\mathbf{v}_t + (\tfrac{1}{2} - \beta)\Delta t^2 \mathbf{a}_t\ ] \tag{5.4}$$

Here $\mathbf{v}$ and $\mathbf{a}$ are velocity and acceleration vectors, respectively, Δt is the specified time increment, t is the time and $\mathbf{x}$ is the increment of displacement response. The Newmark parameters are chosen such that $\beta \geq (\tfrac{1}{2} + \gamma)^2/4$ with $\gamma \geq \tfrac{1}{2}$. The discretization in time are

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t\,\mathbf{v}_t + +\tfrac{1}{2}\Delta t^2[(1-2\beta)\mathbf{a}_t + 2\beta \mathbf{a}_{t+\Delta t})] \tag{5.5}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t(1-\gamma)\mathbf{a}_t + \gamma \Delta t\,\mathbf{a}_{t+\Delta t} \tag{5.6}$$

The object is to solve (5.2) without forming the factors of $\mathbf{A}$.

A splitting method similar to the one used for equation (4.1) can now be applied to equation (5.1). The matrix resulting from the splitting algorithm can then be used as a preconditioner for (5.2). Consider

$$\mathbf{P} = (\mathbf{L} + \frac{1}{\beta \Delta t^2}\mathbf{M})\mathbf{M}^{-1}(\mathbf{L}^T + \frac{1}{\beta \Delta t^2}\mathbf{M}) \tag{5.7}$$

where $\mathbf{K} = \mathbf{L} + \mathbf{L}^T$. Multiplying out the terms in (5.7), we obtain

$$\begin{aligned} \mathbf{P} &= \frac{1}{\beta \Delta t^2}[\beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T + \mathbf{L} + \mathbf{L}^T + \frac{1}{\beta \Delta t^2}\mathbf{M}] \\ &= \frac{1}{\beta \Delta t^2}[\beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T + \mathbf{A}] \\ &= \frac{1}{\beta \Delta t^2}[\mathbf{E}(\Delta t^2) + \mathbf{A}] \end{aligned} \tag{5.8}$$

where $\mathbf{E}(\Delta t^2) = \beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T$.

The preconditioned conjugate gradient algorithm of section 2 is invariant under the scaling of the preconditioning matrix. Therefore, (5.8) shows that $\mathbf{P}$ will tend quadratically to the dynamic stiffness matrix $\mathbf{A}$ as the time step diminishes. In other words, $\mathbf{E}$ tends to the zero matrix quadratically in Δt. We see later that this characteristic results in an effective preconditioning and the solution of equation (5.2) is obtained in as few as 2 or 3 iterations of the preconditioned conjugate gradient algorithm with moderately small Δt.

6. Numerical Examples

In the following, we present a few numerical examples to illustrate some of the characteristics of the proposed preconditioning matrices for the PCG algorithm. This algorithm is implemented in FEAP, a Finite Element Analysis Program (see [8], chapter 24 for more details). All the numerical tests were carried out on a VAX 11/780 at the University of California, Berkeley, using double precision computation.

We first present the results to some static analyses, both linear and non-linear. In these examples we choose a stopping criterion based on the residual vector and the algorithm is terminated as soon as the norm of this vector is reduced by a factor smaller than a specified tolerance. In our calculations we set the tolerance to 10^{-6}. Next we demonstrate our algorithm on a few dynamic problems. The termination criterion is similar to the static case with a range of different tolerances to demonstrate the effectiveness of the algorithm.

6.1 Static Examples: Linear elastic

a) *132 degree-of-freedom building*

The object of the first problem is to determine the influence of the preconditioning parameter, ω, in eq. (4.13). The total number of PCG iterations required to achieve convergence, varies considerably with ω. To illustrate this dependence, we chose the example model shown in figure 1 which is a 132 degrees of freedom, multistory building, discretized by 176 2-node truss elements each with the same Young's modulus (30×10^{6}). The cross-sectional area of the girders, columns and diagonals are 20, 40 and 1 respectively. A single load at the top is applied, as shown in figure 7.

In figure 2, we indicate the number of PCG iterations needed to converge as a function of the preconditioning parameter. The shape of this curve is characteristic of the proposed PCG algorithm and consists of three zones:

1) *Small ω:* The preconditioning matrix approaches the diagonal matrix $\mathbf{D}$. In this case, the total number of iterations is less than that for diagonal preconditioning.

2) *Optimum ω:* With this value the algorithm takes the least number of iterations to obtain the solution. Note that the curve is quite flat around ω_{opt} and therefore the total number of PCG steps is insensitive to small changes in the value of ω. Further, as predicted before, the optimum ω is close to unity.

3) *large ω:* In this range the preconditioning matrix approaches $\mathbf{LM}^{-1}\mathbf{L}^T$ which is a singular matrix (diagonals of $\mathbf{L}$ are zero). In this example with $\omega > 3.0$ the solution may loose accuracy in all significant digits and eventually floating-point overflow occur.

Figure 3 shows the evolution of the residual norm, $\|\mathbf{r}_i\|$, normalized versus $\|\mathbf{r}_0\|$, at the i-th iteration of the PCG algorithm. The residual at each iterate exhibits characteristics typical of conjugate gradient method. Namely, residual norm remains large for a relatively large number of steps before convergence occurs to the specified tolerance. Part of this behavior is due to the loss of orthogonality among the conjugate vectors. Poor preconditioning can also contribute to slow convergence.

b) *"Cantilever beam" type structures*

From the insight we have gained with the preceding example, we now proceed to answer the following question: How to select the ω_{opt}?

No easy analytical solution can be obtained to this question; ω_{opt} depends on the spectrum of $\mathbf{A}$ which is not known *apriori*. However an initial estimate of unity as indicated in section 4 is not an unreasonable choice for ω. The numerical test here is to investigate the dependence of the number of iterations on ω. An accurate upper bound to the total number of PCG steps can be obtained if the condition number of the preconditioned matrix, $\mathbf{P}^{-1}\mathbf{A}$ is known. However the condition number of $\mathbf{P}^{-1}\mathbf{A}$ depends on ω.

The examples we have chosen are summarized in figure 4. Each problem is computed with a range of ω's to obtain ω_{opt}. Figure 5 shows the number of iterations as a function of ω, for these examples. Notice that all the curves are rather flat when close to ω_{opt}, moreover, that ω_{opt} is close to 1.0.

The following table (1) gives the number of iterations for both ω_{opt} and $\omega = 1.0$.

No. of D.O.F. N	ω_{opt}	No. of Iter. for ω_{opt}, k_1	No. of Iter. for $\omega = 1.0$, k_2	$\frac{k_2}{k_1}$	$\frac{k_2}{N}$
30	1.0	14	14	1.00	0.47
40	1.0	23	23	1.00	0.57
60	1.0-1.3	20	20	1.00	0.33
132	1.3-1.4	37	45	1.22	0.28
160	1.25	33	35	1.06	0.21
240	1.0	199	199	1.00	0.83
300	1.2-1.5	46	50	1.09	0.15

Table 1. Comparison of the Number of PCG Iterations for Various ω's.

The last column of Table 1 shows the ratio of total number of iterations over the number of degree-of-freedom. As expected this ratio remains below unity. The next to last column shows the loss in optimality when using ω equal to unity. Except for the 132 degree-of-freedom system little loss in computational effort results from using ω equal to one.

6.2 Static Examples: Nonlinear Elastic Problem

a) *nonlinear material problem*

In much the same way as the Newton-Lanczos method [4], the PCG algorithm was implemented within a Newton loop. The resulting algorithm possessed all the properties of the Newton-Lanczos algorithm with the exception that it is restricted to positive definite matrices. Simplicity of programing various alterations was the motivating factor in restricting attention to the PCG algorithm. Our primary objective is to compare the PCG algorithm with Newton and modified Newton strategies. For this comparison we use the 132 degree-of-freedom truss building described above, but modified to have the same cross-section for all the members ($A = 20$). Nonlinearity is introduced by a simple yield model in the constitutive equation.

A single load is applied at the top with sufficient magnitude to produce a nonlinear maximum displacement of about twice the maximum elastic one. Figure 6 shows the mesh, the deformed structure and the constitutive equation adopted. In Table II, we indicate the relative computational cost comparisons for different methods. We modified all the algorithms mentioned above to include a line search. This was initially expected to reduce the final cost of the algorithms; in fact, for this problem the three methods were more expensive when a line search was included.

Looking at the results in Table II, it is interesting to note that the PCG algorithm required only one more nonlinear step than the Newton method. Also, due to the fact that only the nonzero terms of the stiffness matrix are stored, the cost for one matrix-vector operation in the PCG algorithm is smaller than for the other methods. For this example the number of terms in the matrix stored in profile form is 1654, however the PCG algorithm requires only 512 nonzero terms. Therefore the cost of one matrix-vector operation is about a third of a profile multiply. More important is the reduction in the over all storage demand. For this example the storage is reduced to 31% of the amount required for a profile stored solution.

Method	No. of iter.	No. of LU factor.	No. of function evaluations	No. of matrix-vec. operations
Newton	6	6	7	6
Mod. Newton	215	1	216	215
PCG	7	0	8	186
Newton + LS	6	6	7	6
Mod. Newton + LS	111	1	112	111
PCG + LS	7	0	8	176

Table II. Cost Comparisons for different Nonlinear Methods (Truss Example).

The average number of PCG iterations was 27, for a preconditioning parameter $\omega = 1.5$. This compares with 37 PCG iterations for a linear problem with much the same structure (see section 6.1a). A lower tolerance for PCG algorithm is used in the earlier stages of the Newton loop which accounts for the lower average number of iterations (see [4] for more details).

In this test the total cost for the *PCG algorithm* was twice the Newton cost. However, this ratio is expected to drop well below 1.0 for three-dimensional structures where the cost of a factorization is large compared to the matrix-vector operation. Moreover, as noted previosly we require substantially less storage space.

b) *finite deformation problem*

In figure 7, we show a plane strain rubber block subjected to large deformation. We employ a 4-node element and a Mooney-Rivlin material as described in

[7]. The rubber block is discretized by 144 elements (12 x 12 mesh) with 286 degrees of freedom. The stiffness matrix stored in profile form requires 7618 storage spaces of which only 1799 are nonzero; corresponding to a 76% saving in storage when using PCG.

The rubber block is stretched to 50% of its original length in load 5 steps. The cost comparison of both the PCG and Newton algorithms is summarized in the following table.

displ.	Nonlinear PCG		Newton
u	No. of Iterat.	Averg. No. Matrix op.	No. of Iter.
0.1	6	43	7
0.2	6	45	6
0.3	7	47	6
0.4	6	47	5
0.5	6	44	5

Table III. Comparision of PCG Algorithm and Newton Method (Rubber Block).

Again, both PCG and Newton require almost the same number of nonlinear steps to converge. What is more interesting is that the number of PCG iterations is quite constant, even for the highly nonlinear range. When comparing this test with the previous 132 degree-of-freedom building example, we notice that the number of iterations in the PCG algorithm, as expected, does not increase as fast as the number of degree-of-freedom.

6.3 Dynamic example

In this example, we wish to indicate the effectiveness of solving approximately, a linear elastic dynamic problem using the PCG algorithm. This is done for a series of time steps and tolerances. Since the PCG algorithm involves no factorization steps it can solve nonlinear dynamic problems with the same amount of effort as for the linear case. However, in this study we select a linear problem. To limit the computer costs, we selected a structure having 20 4-node plane stress elements defined in section 6.1(b). The dynamic problem consists of releasing the structure from an initially deformed configuration and letting it vibrate freely. The mesh, material properties and initial state are given in figure 8. The time steps chosen are Δt = 0.5, 0.2, 0.1, 0.05 and 0.025 seconds, which correspond to $1/120 < \Delta t / T < 1/6$, where $T = 3.0$ sec. is the fundamental period of vibration of the structure. For comparison, the critical time step for an explicit analysis would be $\Delta t_{cr} = 0.01$ sec., for a bulk wave velocity of 911/s. In order to see the effect of solving approximately the set of equations, we use the three following tolerances: $tol = 10^{-4}$, 10^{-2} and 10^{-1}. In figure 9, we plot the y-displacement of node 1 for the time step $\Delta t = 0.1$ for the three tolerances; in figure 9, we show the corresponding relative error e, i.e.

$$e = \frac{\delta_N(t) - \delta_{pcg}(t)}{\delta_0}$$

where δ_N is the displacement obtained using the Newmark method, δ_{pcg} is the corresponding results obtained using PCG and $\delta_0 = 0.172$ is the initial applied displacement. The results clearly show that the tolerance 10^{-1} is too large and leads to inaccurate results. The error using a tolerance of 10^{-2} is about 1%, while there is no visible error for $tol = 10^{-4}$ (less than 0.01 percent). When we reduce the time step to $\Delta t = 0.05$ (half the preceding), the results improve substantially: while we see no difference between tolerances 10^{-4} and 10^{-2}, there is only 1% error when using 10^{-1} (fig. 10). For smaller time steps, no differences are seen in the first five digits.

Finally, figure 11 shows the average number of iterations as a function of $T/\Delta t$. The reduction of the number of iterations as Δt tends to zero is quite interesting: for a tolerance of 10^{-2}, this number drops from 30 to 4 iterations when $\Delta t / T$ changes from 1/6 to 1/120. This reduction is totally due to the convergence of the preconditioning matrix to the dynamic stiffness matrix, $\mathbf{A}$. Such a small time step is not unusual in many applications, e.g. impact problems.

7. Closure

In this report we have described our initial efforts to construct a solution method for the algebraic equations arising from finite element solution of linear and non-linear problems. Both static and dynamic problems are considered. For nonlinear problems, Newton's method is used to generate a sequence of linear problems. A preconditioned conjugate gradient method is used to solve the linear set of equations. A method for constructing an effective preconditioning matrix in terms of an additive decomposition of the coefficient matrix is introduced separately for the static and dynamic cases. Several example problems are solved demonstrating the features of the proposed method.

In order to further evaluate the method additional work is required. In particular we recommend that the conjugate gradient part of the algorithm be replaced by the Lanczos method as described in [4,6]. This will permit consideration of indefinite problems, such as those resulting from use of Lagrange multiplier methods (e.g.,contact problems, etc.). In addition it is essential to test the method on larger problems, preferably some three-dimensional problems where sparsely populated coefficient matrices with rather large mean column heights occur. Further analyses for significant non-linear problems should also be performed. Finally, some efforts to adaptively compute an optimal value for ω should be explored.

REFERENCES

[1] Concus, P., G. H. Golub and D. P. O'Leary, "A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations," *Sparse Matrix Computations*, ed. by J. R. Bunch and D. J. Rose, Academic Press, New York, 1976.

[2] Kershaw, D. S., "The Incomplete Cholesky-Conjugate Gradient Method for Solution of System of Linear Equations," *Journal of Computational Physics*, vol. 26, pp. 43-65, 1978.

[3] Gourlay, A. R., "Splitting Methods for Time Dependent Partial Differential Equations," *The State of the Art in Numerical Analysis*, ed. by D. Jacobs, Academic Press, New York, 1977.

[4] Nour-Omid, B., B. N. Parlett and R. L. Taylor, "A Newton- Lanczos Method for Solution of Nonlinear Finite Element Equations, *Computers and Structures*, vol. 16, No. 1-4, pp. 241-252, 1982.

[5] Park, K. C. and J. M. Housner, "Semi-Implicit Transient Analysis Procedures for Structural Dynamic Analysis," *International Journal for Numerical Methods in Engineering*, vol. 18, pp. 609-622, 1982.

[6] Parlett, B. N., "A New Look at the Lanczos Algorithm for Solving Symmetric Systems of Linear Equations, *Linear Algebraic Applications*, vol. 29, pp. 323-346, 1980.

[7] Simo, J. C. and R. L. Taylor, "Penalty Formulations for Rubber-like Elasticity, *Report No. UCB/SESM 81/03*, Department of Civil Engineering, University of California, Berkeley, November 1981.

[8] Zienkiewicz, O. C., *The Finite Element Method*, Third Edition , Mc-Graw Hill, Inc., London, 1977.

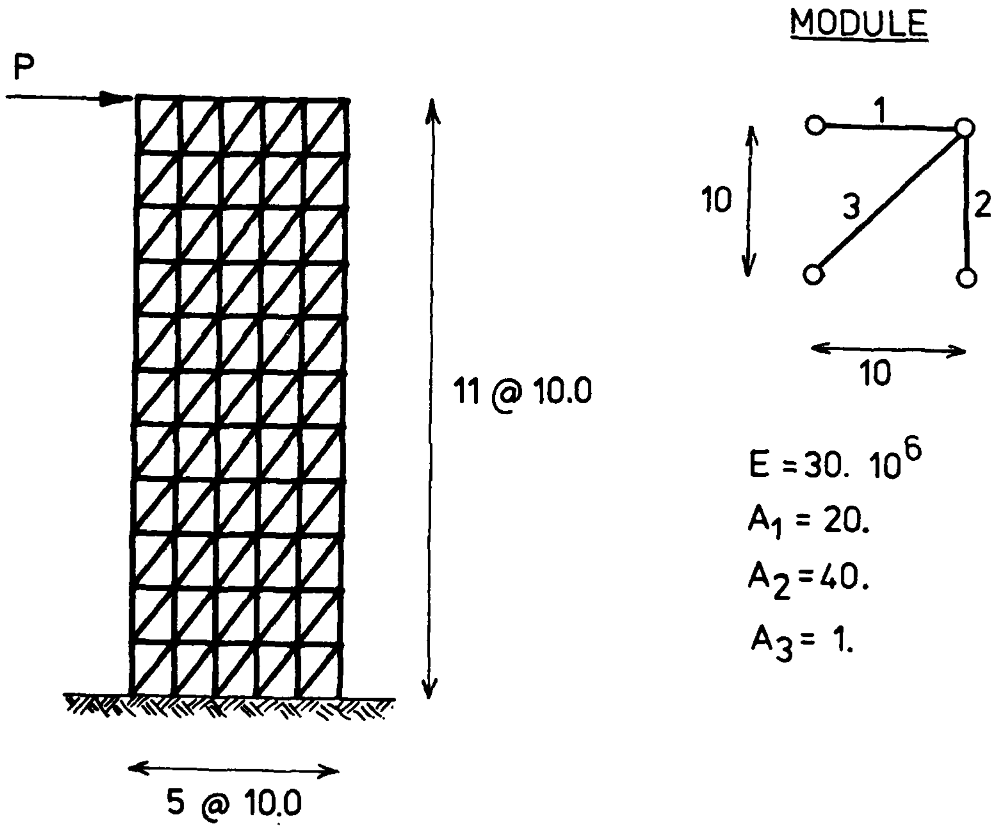


Figure 1. 132 Degrees of Freedom Truss Building.

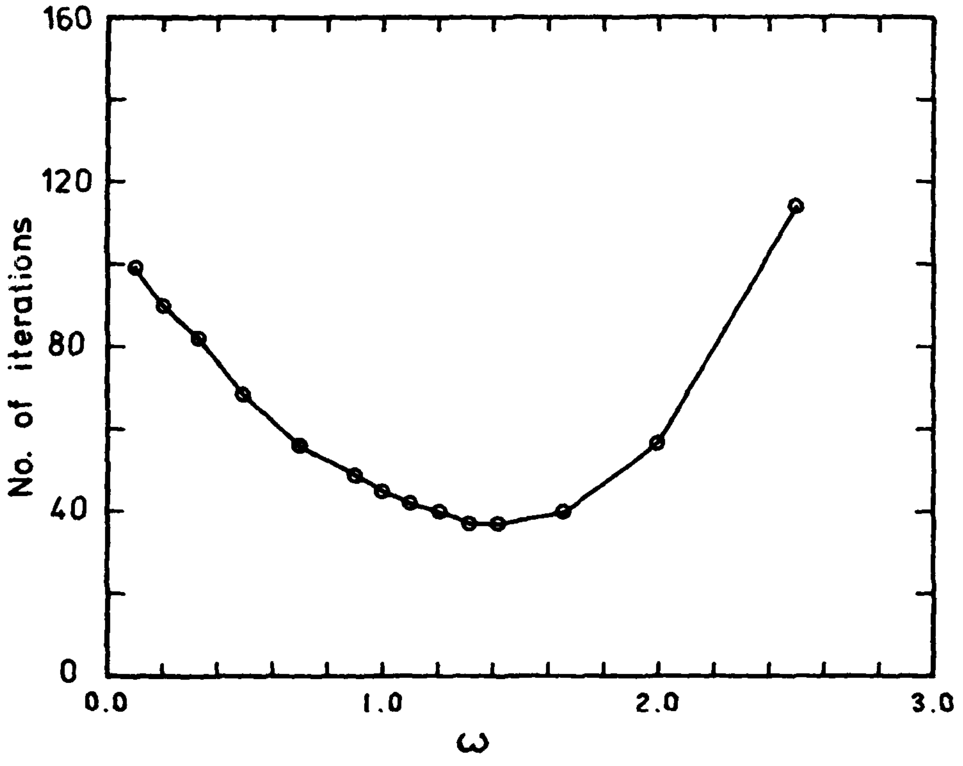


Figure 2. Effect of ω on the Number of Iterations.

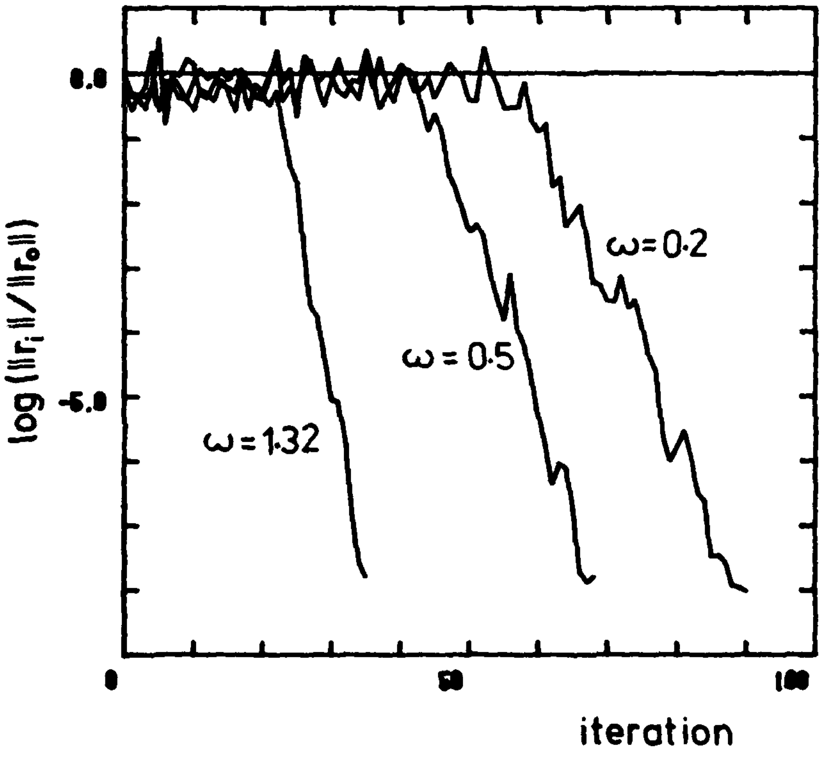


Figure 3. Reduction of the Residual Norm for PCG Iterations.

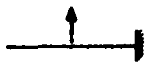
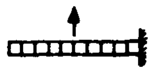
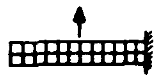
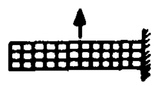
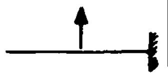
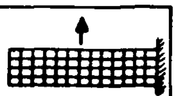
GEOMETRY						
ELEMENT TYPE	10 BEAM ELEMENTS	10 PLANE STRESS ELEMENTS	20 PL. ST. EL.	60 PL. ST. EL.	80 BEAM EL.	120 PL. ST. EL.
NUMBER OF DOFS	30	40	60	160	240	300

Figure 4. Summary of the Set of Examples Chosen for the Static Tests.

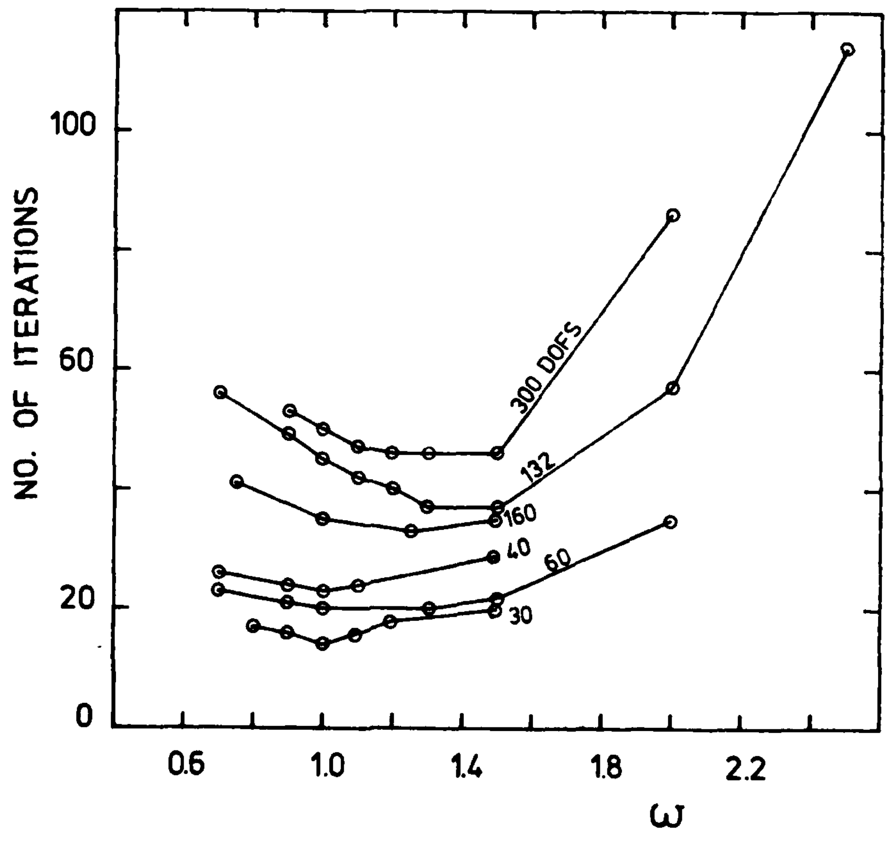


Figure 5. Variation of the Number of Iterations with ω.

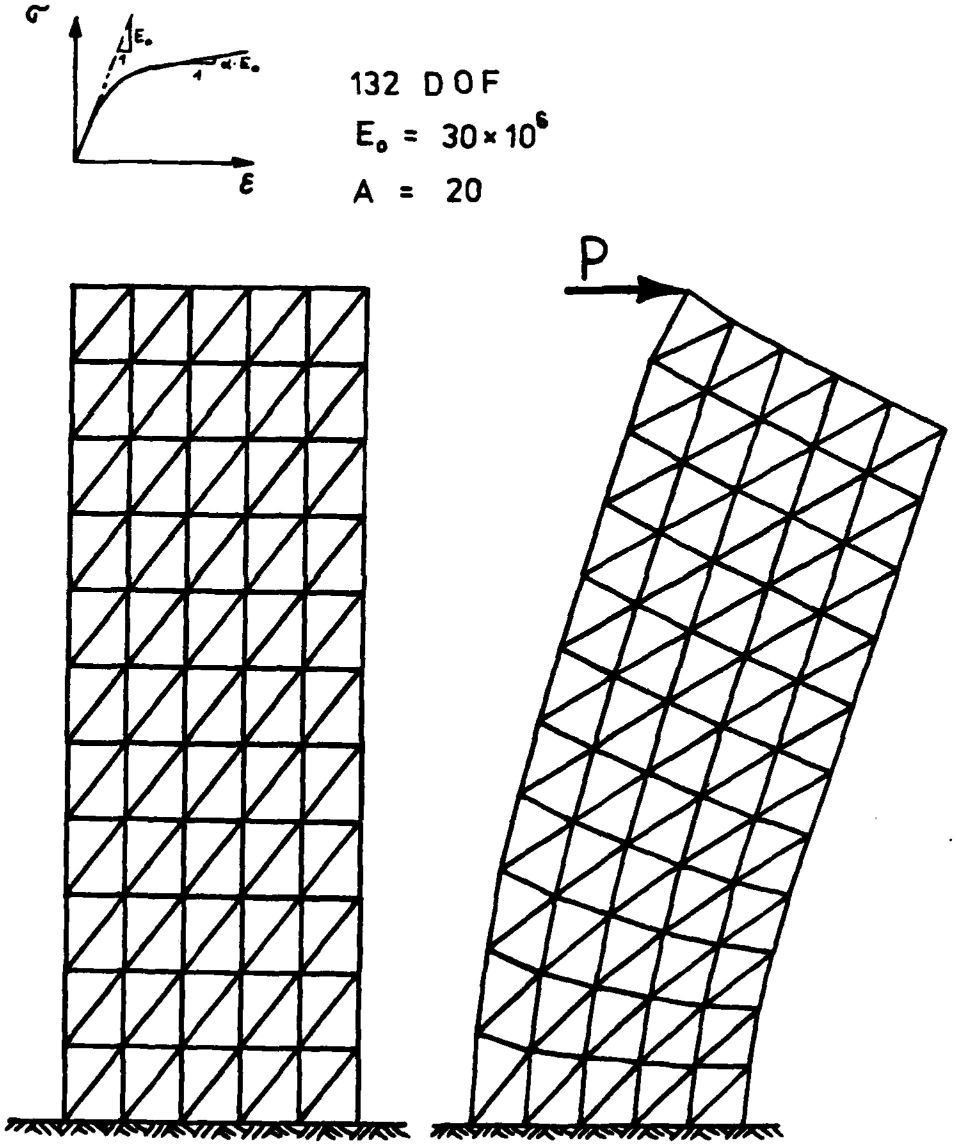


Figure 6. 132 Degrees of Freedom Nonlinear Truss Building.

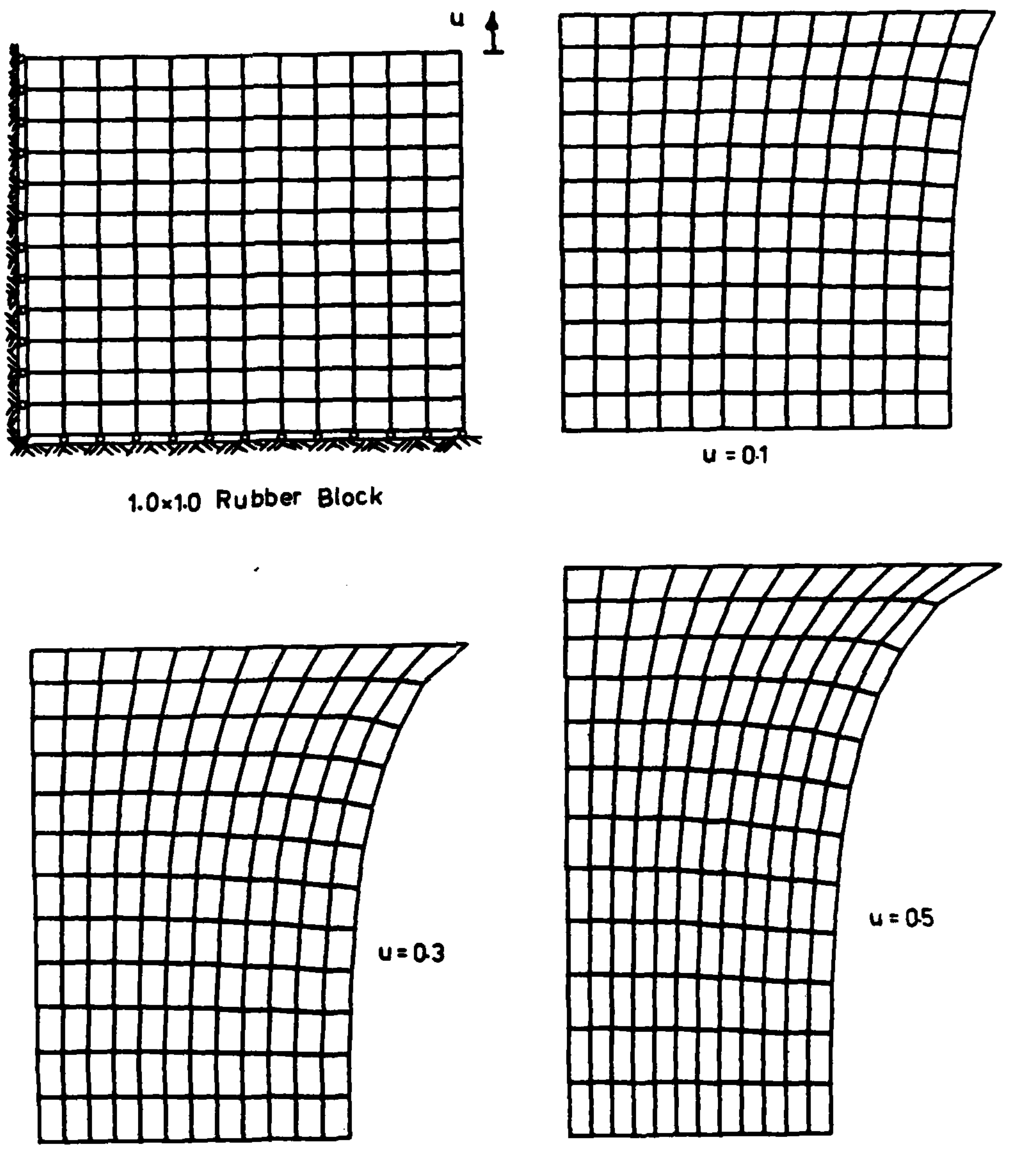


Figure 7. Large Deformation Analysis of the Rubber Block with Mooney-Rivlin Material Model.

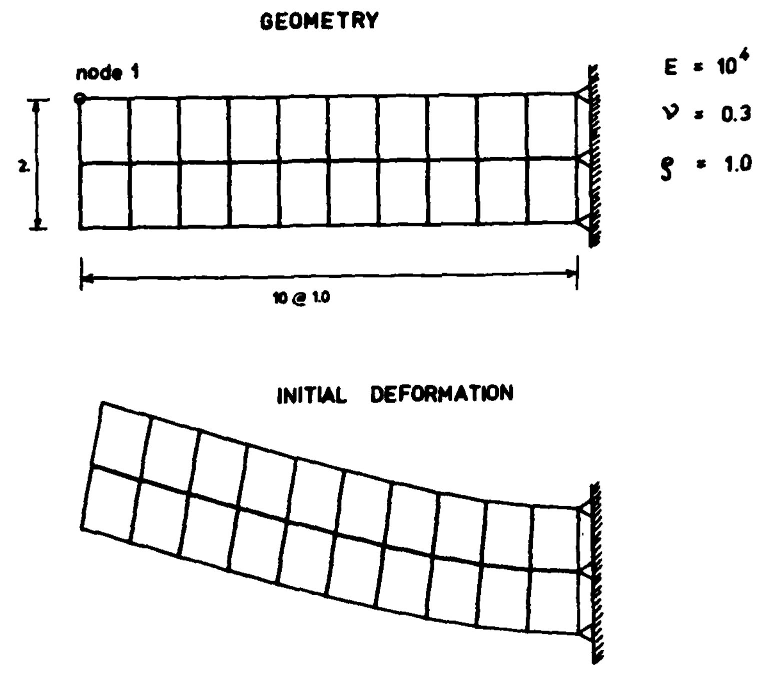


Figure 8. Beam Example Used in Dynamic Analysis.

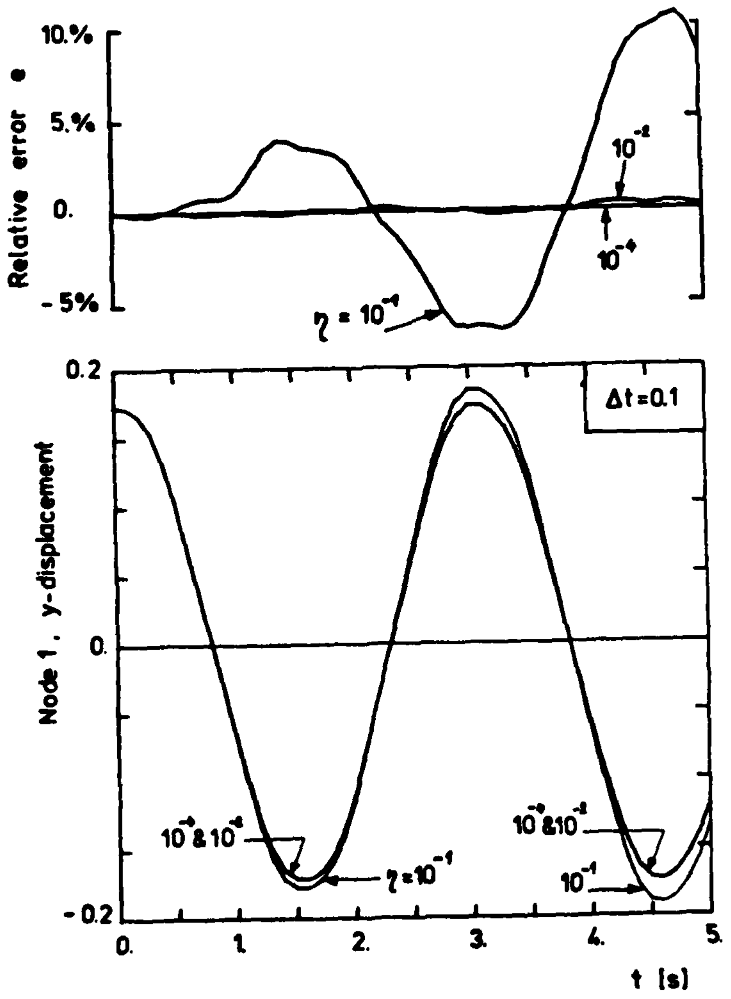


Figure 9. Response of Node 1 with Different PCG Tolerances and Large Time Step.

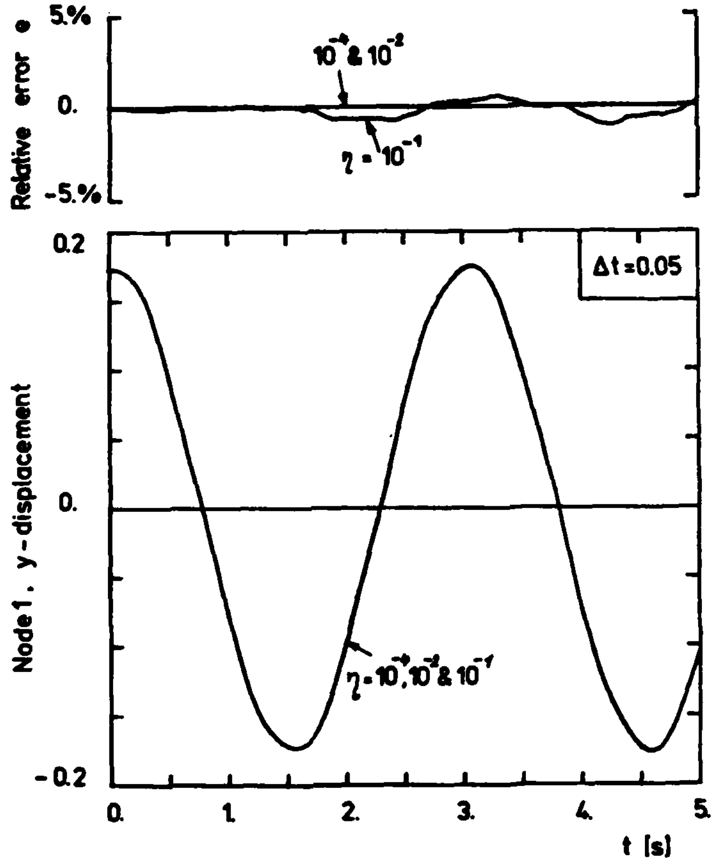


Figure 10. Response of Node 1 with Different PCG Tolerances and Small Time Step.

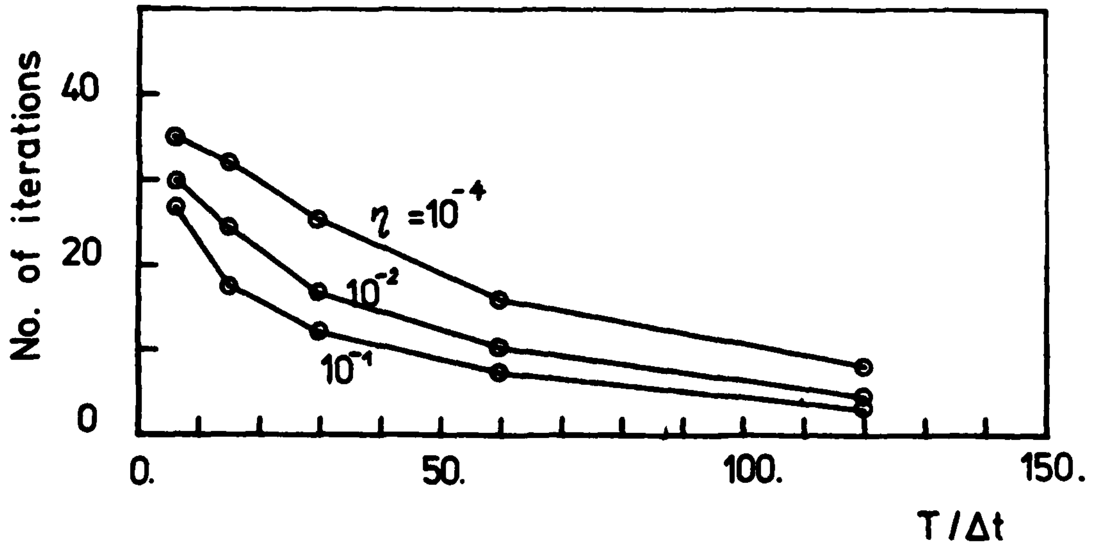


Figure 11. Improvement of Preconditioning Matrix with Reduction in Time Step.